**X00202628 – Vlad Pocris – CA3 Report**

**Breach Explorer   
URI - https://gcstorageacc2.z9.web.core.windows.net/  
GitHub -** [**https://github.com/VladPocris/CA3\_EAD1**](https://github.com/VladPocris/CA3_EAD1)

**Video - https://youtu.be/e0jaUCouGFw?si=hfn4gVaTCz-fJkKD**

**Descritpion**

This application is about data breaches from big companies website, it provides users with information about all data breaches that occurred using 3 API’s from Have I Been Pwned? Website.

It’s a interactive website that lets you visualise the data retrieved from a database that is regularly updated. For visualization a bar chart is used highlighting number of data that has been stolen e.g. email addresses, passwords, usernames and more. The website is responsive and partially compatible with mobile devices, 85% of the application is designed to fit small screens some of the text still haven’t been designed properly for devices with very small screen. The webpage starts with an animation aka slogan, which has a typing animation. After the slogan the website has 2 components which are the Latest Breach container highlighting the latest breach occurred showing details, description and if it is a officially verified breach or not. Then the user can see a visualization that he can change based on it’s preference, two buttons are available for making the chart horizontal or vertical showcasing the top 15 biggest breaches of all time. Next the user can use a custom bar chart that at the start is empty, there is a input where the user can search for breached domains and also gets suggestions, the chart has buttons that allow to user to either add/remove/clean or add random data to the visualization. The next page Was I Breached welcomes the user with a title and an input where the user could specify the email he want’s to check to see where the email has been exposed and if there is any recorded data then the user gets in how many breaches it was exposed and information about the breaches, if nothing is found then the user gets a congratulations message. This page uses an API that does not have CORS enabled, however I did make that possible by creating a proxy API a workaround which is published on my Azure account which has been set up to allow CORS, this way I was able to effectively retrieve the names of the data breaches, after that another API has been used that would simply return the data about breaches based on the breaches names retrieved from the first API.  
The website also includes some SVG manipulations used to create animations, for example the logo on the top of the navigation bar gets coloured in the blue violet colour when you hover.

**Code-Description**

Home.razor

GetBreachAsync() and GetBreachesAsync() are the API calls that I configured to retrieve data.  
GetBreachAsync() – uses LatestBreach Have I Been Pwned API V3, it does not require an API-key so is simply an endpoint that retrieves json data.  
GetBreachesAsync() – uses All Breaches endpoint that retrieves all breaches from the database.  
Classes are made with help of the json to c# website, very straightforward.

In these Tasks I also use a variable called success that lets the website load only when all the data is retrieved first, the loading screen is also changed using a custom gif animation with a transparent background.   
GetPAsswordAsync() – async Task that retrieves a password using ninja-apis endpoint for password generator using 3 parameters, length, exclude\_numbers, and exclude\_special\_chars, this one require an api-key.

UpdatePassword() – handles the button click calling the GetPasswordAsync(), assigning a value to a variable to make the copy button visible that is binded for a style.  
  
UpdateTooltipValue() – basically updates the tooltip that is used to set the length for the length parameter, mainly it updates the style so you can move it around and also assign the value to the input.  
  
CopyTextToClipboard() – handles the click of the button that is visible after the button for UpdatePassword() is clicked, I couldn’t find a way to copy it using blazor, some sources stated that you can’t do that in blazor so I had to use a function in js, so basically what this function does is it invokes the function from the js file and shows a tooltip confirming that the password has been coped.  
  
GetLabelsTop() – gets the names of data breaches from the api call made before in order to be used for chart initialization.  
GetDataTop() – gets the pwned count (breached data - quantity) from the same api as for the last one.  
Both functions uses Linq to select the top 15 breaches.  
InitializeChartDataTop() – initialization of the data chart, using the documentation from blazor bootstrap samples I just simply create a dataset, assign the colours, and set the options the way I want it to be, and at the end I just call an initialization method for the bar chart object passing the dataset and options.  
ShowHorizontalBarChartTopAsync() and ShowVerticalBarChartTopAsync() changes the options and just call the UpdateAsync() function with passed parameters and updates the chart. Also got from documentation.  
  
For the custom chart is mainly the same exact format as for the top 15 breaches the only difference is that I had to create different async Task’s that would change the dataset and options and then simply call the UpdateAsync() passing the new datasets and options.  
It also uses ToastService a component from the BlazorBootstrap that let’s me notify the user in different circumstances, the code is very well architected since it addresses all the validation, e.g. if the user doesn’t specify a value and then presses enter then a toast message on the right top corner appears stating that “Please specify the company name to add.”. Very handy feature from blazor’s bootstrap.  
  
I won’t go into all the details of the code, since I believe it is self explanatory but at least I could give an overview of what they do. If there are any questions about it please feel free to ask, I understand all the code and everything has been written by me, therefore I could tweak it on request.  
  
Breached.razor  
GetBreachNamesAsync();

GetBreachAsync();

The only one I will explain is the GetBreachAsyn();  
It gets the output from GetBreachNamesAsync() which is a list, and for each element it makes API calls to get full details using each breach names which are then displayed on the page.  
One important thing to notice is that <https://breachproxy.azurewebsites.net/api/breaches/breachedaccount/> **is a proxy API created by me that I published on Azure so I could use an API that has no CORS enabled. This page should work until 16 December I will provide some screenshots so you know that it works now in case you pass 16th of December, it is a subscription I got since it looked like an interesting thing to work with.**

Also playwrite tests also are written which checks the main functionalities of the websites such as navigation bar, home page and breach page features, the one I couldn’t test is the charts, everything else that was interavtive has been checked e.g. password length, including numbers or special characters and different scenarios, also the tests checks the results the API retrieves if it found something about the users email or not. I won’t go into details since it was really easy, mainly just assert and access the elements of the website.

**Screenshots  
HOME-PAGE**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated **(Trying to remove an unexistent company from the chart).**

A screenshot of a computer

Description automatically generated  
Generating a password.  
**BREACHED-PAGE**A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

**Code Quality Metrics**The code could be definitely improved, I’ve been not consistent in using the ways I handle some validation but I wanted to experiment. Also some complexity there could’ve been either made simpler or either split in different method, especially with the custom chart, could’ve been a better and less code approach. Overall the performance of the website is good, and I did follow the main naming conventions.